

APPENDIX A

```

#!/usr/local/bin/perl

5  # Copyright (c) 2001
  # LSI Logic, Inc.
  #
  # Andy Rankin
  # Ph: (970) 206-5107
10 # email: arankin@lsil.com
  #
  # sp2vlog.pl
  #
  # Usage: sp2vlog.pl -s spfile [-d dfile] [-h]
15 #
  #       -s <spfile>   Spice transistor level file to be
  translated (required)
  #       -d <dfile>    Data file from vlog2sp.pl containing port
  types (defaults to basename.data)
20 #       -h            prints this message
  #

  # ADD CLEANER TO THIS FILE TO GET + SIGNS REMOVED
  # ADD SOMETHING TO VLOG2SP TO MAKE SPICE COMPATIBLE LINE LENGTHS
25 #####
  # Set Defaults #
  #####

30 $tech="gflx";
  $subckt_lead_string="x";

  #####
  # Parse Command Line Arguments #
35 #####

  require "newgetopt.pl";
  &NGetOpt("s:s","d:s","h");

40  if ($opt_h) { &usage; }
  if ($opt_s) {
    $spfile=$opt_s;
  }
  else {
45    print("\nERROR: Must specify spice file\n");
    &usage;
  }

  $basename=`basename $spfile .sp`;
50 chop($basename);

```

```

if ($opt_d) { $dfile=$opt_d; }
else { $dfile=$basename.".data"; }

#####
5 # Open spice and data files #
#####

open(INFILE,"$spfile") || die("Unable to open $spfile for
reading: $!\n\n");
10 @infile=<INFILE>;
close(INFILE);

open(INFILE,"$dfile") || die("Unable to open $dfile for reading:
$!\n\n");
15 @dfile=<INFILE>;
close(INFILE);

foreach (@infile) {
    @pin_list=();
    tr/A-Z/a-z/;
    if (/^\s*.subckt\s/) {
        # If the line is a spice subckt definition, it is the
        same as a module definition in
        # verilog. Split the line to extract the subcircuit
        name and pins:
        chop;
        ($foo,$module,@subckt_pins)=split(/\s+/);
        @module_pins=&remove_busses(@subckt_pins); # remove
        bus notation from subcircuit pins
        $pinlist=join(" ", @module_pins);
        30 print ("module $module \( $pinlist \)\;\n"); # print
        the verilog module definition line

        # For each pin on the module, determine the port type
        35 from the data file:
        foreach $pin (@module_pins) {
            foreach $port (@dfile) {
                if ($port =~ /$module $pin (\S+)/) {
                    $port_type=$1; last; }
                    else { $port_type=""; }
                }
                @indices=&consolidate_bus($pin,@subckt_pins); #
                If it's a bus, add bus notation

                # Print the input/output definitions with bus
                notation as necessary:
                if ($port_type) {
                    if (@indices) { print("$port_type
                    [$indices[$#indices]:$indices[0]] $pin\;\n"); }
                    else { print("$port_type $pin\;\n"); }
                    }
                }
            }
        }
    }
}

```

```

    }

    # Print wire definitions for all the internal signals:
    foreach (@dfile) {
5       if (/ $module (\S+) wire/) { print("wire $1\;\n");
    }
    }

    }
10    elsif (/^\s*.ends\s*/) {
        print("endmodule\n\n");
    }
    elsif (/^\s*x/) {
15        s/^\s+//;
        s/\s+$//;
        @signal_names=split(/\s+/);
        $instance_name=shift(@signal_names);      # Shift the
instance name from front of line
        $instance_name=~s/^\$subckt_lead_string//; # Remove the
20    X from the
        $subckt_name=pop(@signal_names);          # Pop the
module name from the end of line
        @signal_names=&clean_signals(@signal_names);

25        #grep the spice file for the subcircuit definition,
and extract the pin names:
        ($foo,$foo2,@subckt_pins)=split(/\s+/,`grep -i
".SUBCKT $subckt_name " $spfile`);

30        # Verify that the number of signals in the spice
subcircuit call matches the number of
        # pins in the spice subcircuit definition:
        if ($#signal_names != $#subckt_pins) {
            die("ERROR: $instance_name $subckt_name number of
35    pins do not match\n\n");
        }

        # Pair up the signal names in the spice subcircuit
call with the pin names
40        # on the spice subcircuit definition:
        for($i=0;$i<=$#signal_names;$i++) {
            $subckt_pins[$i]=~tr/A-Z/a-z/;

            push(@pin_list,".$subckt_pins[$i]\($signal_names[$i]\)");
45        }

        # Join the list of signal/pin pairs and print it with
the module instantiation:
        $pin_string=join(" ",@pin_list);
50        print("$subckt_name $instance_name ( $pin_string
)\;\n");

```

```

    }

5   }

    sub usage {

        # This subroutine prints usage information

10      print("\nusage: sp2vlog.pl -s spfile [-d dfile] [-h]\n\n");
        print("\t-s <spfile>\tspice transistor level file to be
translated (required)\n");
        print("\t-d <dfile>\tdata file from vlog2sp.pl containing
port types (defaults to basename.data)\n");
15      print("\t-h\t\tprints this message\n");
        die("\n");
    }

    sub consolidate_bus {

20      # This subroutine finds the indices for a given pin, sorts
        them, and returns them.

        my($pin,@pins) = @_;
25      my(@indices);

        foreach (@pins) {
            if (/ $pin\[ (\d+) \]/) {
                push(@indices,$1);
30          }
        }
        @indices=sort(@indices);

35    }

    sub remove_busses {

        # This subroutine removes bus notation from the pins on the
        spice subcircuit line.
40      # It includes each bus as one pin in the pinlist. It
        returns the pin list.

        my(@pinlist) = @_;
        my(@newpinlist);

45      foreach (@pinlist) {
            s/[ \d+ ]//g;
            if(&element_exists($_,@newpinlist)) {}
            else { push(@newpinlist,$_); }
50      }
    }

```

```

        return(@newpinlist);
    }

    sub element_exists {
5         # This subroutine checks to see if an element exists in an
        array.

        my($element,@array)=@_;
10        foreach (@array) {
            if ($_ eq $element) { return(1); }
        }

15        return(0);
    }

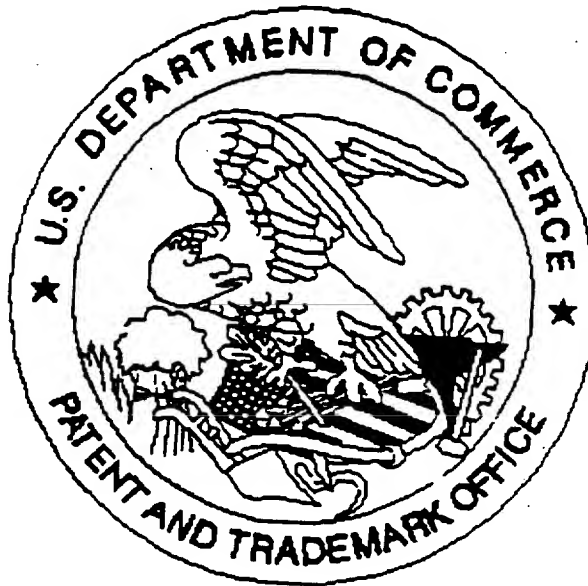
    sub clean_signals {

20        # This subroutine converts vss and vdd signals to verilog
        logic 0 and logic 1. It also
        # removes dummy pins.

        my(@signals) = @_;
25        foreach (@signals) {
            if ($_ eq "vss") { $_ = "1'b0"; }
            if ($_ eq "vdd") { $_ = "1'b1"; }
            if (/^dummy/) { $_ = ""; }
30        }
        @signals;
    }

```

United States Patent & Trademark Office
Office of Initial Patent Examination – Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

* pages numbered 14 - 18 as part of specification are
Appendix.

☐ Scanned copy is best available.

SCANNED # 244